JURNAL TEKNOSAINS

ISSN 2089-6131 (Print)
ISSN 2443-1311 (Online)
DOI 10.22146/teknosains.11380
https://jurnal.ugm.ac.id/teknosains

VOLUME 6 No. 1, 22 Desember 2016 Halaman 1-58

APPLYING ALPHA-BETA ALGORITHM IN A CHESS ENGINE

Werda Buana Putra **dan** *Lukman Heryawan*
Computer Science Department
Universitas Gadjah Mada
Email: magmaflood@gmail.com

ABSTRAK

Algoritma Minimax dikenal sebagai solusi untuk mengurangi beban pencarian pada mesin catur. Namun, metode yang lebih dalam diperlukan untuk meningkatkan performa algoritma ini. Salah satu solusinya dikenal dengan algoritma Alpha-Beta Pruning. Idenya adalah dengan mengeliminasi cabang yang dianggap tidak diperlukan di dalam pohon pencarian.

Kata Kunci: ***Algoritma; Alpha-Beta Pruning; Mesin catur; Minimax; Pohon pencarian.***

ABSTRACT

Minimax Algorithm, is a solution to reduce the burden on hardware in chess engine. However, a more in-depth method is needed to further increase the search algorithm. One of those solutions is called Alpha-Beta Pruning algorithm. The idea is to eliminate the unnecessary nodes in the search tree.

Keywords: ***Algorithm; Alpha-Beta Pruning; Chess engine; Minimax; Search Tree.***

INTRODUCTION

The origin of chess spans over 1500 years ago. The earliest predecessor of the game probably originated in India. From India, the game spread to Persia. When the Arabs conquered Persia, chess was taken up by the Muslim world and subsequently spread to Southern Europe. In Europe, chess evolved into roughly its current form in the 15th century.

Chess

Chess is one of the most well-known 2-player board game. Played on an 8x8 white and black squares. One player playing white pieces, while the other playing as black pieces

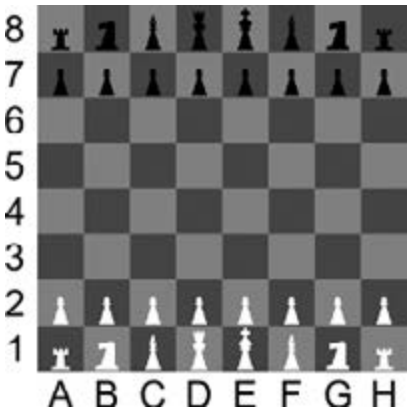

Figure 1.
Chess game Initiation

When the game starts, each player put the 16 of their chess pieces on the board in a specific order. These chess pieces will be moved to attack, defend, threat, and capture the other player's pieces. The difference in their characteristic is the main factor while constructing strategies in chess.

The aim of the game is to capture the opponent's 'King' piece. The game will end either when a king is captured or when the condition for stalemate has met. Stalemate refer to as draw, the condition for this are determined by FIDE, a World Chess Federation.

Chess Pieces and Their Move Range

Each player would be given 16 pieces, with 8 different role of pieces consisted of 8 pawns, 2 knights, 2 bishops, 2 rooks, a queen and a king with each role has their own unique move range. One thing in common is that the move path cannot be looped on the board. Meaning that when a piece has reached a certain corner of the chess board, they must stop and they have to go all the way back if they want to reach another side. *Firstly,* Pawn: Pawns are allowed to move 1 square forward toward enemy zone. On their first move, they are allowed to move 2 squares forward, however, they captures by moving 1 square diagonally forward. And should they be able to reach the last row of opponent's side, pawn can be promoted into any piece. *Secondly,* Rook: Rook are allowed to move straight as far as the board allows it. *Thirdly,* Bishop: Bishop are allowed to move diagonally in any range. *Fourthly,* Knight: Knight's move are different than any other pieces, they move in an "L" letter shape with total of 3 square move. *Fively,* Queen: The most powerful piece of all. Queen move range is the combination of bishop's and rook's move range. *Sixthly,* King: Similar to Queen, only that King can only move 1 square to it's neighbor squares.

Setting Up the Board

The fundamental of building a chess engine is the representation of the chess board which affecting on how the chess engine will track the board and observe the ruling. Our chess engine, which we will refer as "Harmonia", is designed as such so it would be able to interract with Arena, a GUI for open source chess engine. It is essential to properly understand what a chess engine required to be able to communicate with the GUI. However, we will skip this part and move to the design. Out of 24 parts which construct "Harmonia", we will talk about 3 of those which is essential. In file board.cpp, we will find out how the engine recognize, not only the whole board, but also all the pieces.

The chess board will be recognized with a representation of 8x8 matrices. The row will be coded from number 1 to 8, and the column will be represented with letter 'a' to 'h'.

```
Int i,type,k=7;
char ch;
fprintf(fout,"   abcdefgh\n +---------\n8| ");
```

Figure 2.
Chessboard Initiation Code

The pieces will be represented with letter that symbolized it's role, it is convenient that c++ is case sensitive, because this way, we will be able to identify the pieces for each side by giving them the same letter, if the pieces role is the same, yet the engine can differ them by using different letter case.

```
type=board;
if(type==0) ch='P';
else if(type==1) ch='N';
else if(type==2) ch='B';
else if(type==3) ch='R';
else if(type==4) ch='Q';
else if(type==5) ch='K';
else if(type==7) ch='p';
else if(type==8) ch='n';
else if(type==9) ch='b';
else if(type==10) ch='r';
else if(type==11) ch='q';
else if(type==12) ch='k'
```

Figure 3.
Naming Chess Pieces

RESULT AND DISCUSSION

Basic Evaluation

File Evaluation. h provide a sequence of how the chess pieces will get value depended on their position. These value, later on, will be used to run Harmonia.

```
// Pawn scores White
{    0,  0,  0,  0,  0,  0,  0,  0,
    20, 26, 26, 28, 28, 26, 26, 20,
    12, 14, 16, 21, 21, 16, 14, 12,
     8, 10, 12, 18, 18, 12, 10,  8,
     4,  6,  8, 16, 16,  8,  6,  4,
     2,  2,  4,  6,  6,  4,  2,  2,
     0,  0,  0, -4, -4,  0,  0,  0,
     0,  0,  0,  0,  0,  0,  0,  0},
```

Figure 4.
Pawn's Basic Value

Pawn's evaluation point is at It's peak just before the last square from it's opponent's side (ie : it would be 7th row for white's pawn). They will get 0 point as they will get promoted immediately into another piece type.

```
// Knight scores White
{   -40, -10, - 5, - 5, - 5, - 5, -10, -40,
    - 5,  5,  5,  5,  5,  5,  5, - 5,
    - 5,  5, 10, 15, 15, 10,  5, - 5,
    - 5,  5, 10, 15, 15, 10,  5, - 5,
    - 5,  5, 10, 15, 15, 10,  5, - 5,
    - 5,  5,  8,  8,  8,  8,  5, - 5,
    - 5,  0,  5,  5,  5,  5,  0, - 5,
    -50, -20, -10, -10, -10, -10, -20, -50,},
```

Figure 5.
Knight's Basic Value

Knight is the only piece which can hop over another piece (s). This particular trait made them as the best support when played in middle field. However, due to it is limited move range, they are most vulnerable in corner side of the board.

```
// Bishop scores White
{   -40, -20, -15, -15, -15, -15, -20, -40,
      0,  5,  5,  5,  5,  5,  5,  0,
      0, 10, 10, 18, 18, 10, 10,  0,
      0, 10, 10, 18, 18, 10, 10,  0,
      0,  5, 10, 18, 18, 10,  5,  0,
      0,  0,  5,  5,  5,  5,  0,  0,
      0,  5,  0,  0,  0,  0,  5,  0,
    -50, -20, -10, -20, -20, -10, -20, -50
},
          // Rook scores White
{   10, 10, 10, 10, 10, 10, 10, 10,
     5,  5,  5, 10, 10,  5,  5,  5,
     0,  0,  5, 10, 10,  5,  0,  0,
     0,  0,  5, 10, 10,  5,  0,  0,
     0,  0,  5, 10, 10,  5,  0,  0,
     0,  0,  5, 10, 10,  5,  0,  0,
     0,  0,  5, 10, 10,  5,  0,  0,
     0,  0,  5, 10, 10,  5,  0,  0,
},
```

Figure 6.
Bishop and Rook's Basic Value

Even tough both rook and bishop has a wide move range, the proportion of their evaluation is very different. For rook's case, they still possess huge threat for opponent at the far end of the board. Unlike rook, bishop has many disadvantages while on the board's edge, either in friendly side or in the opponent's side.

It's so because while in corner, bishop only have 1 way move range. This is the reason as to why most player value bishop at the same value as a knight.

```
// Queen scores White
{   0,  0,  0,  0,  0,  0,  0,  0,
    0,  0,  0,  0,  0,  0,  0,  0,
    0,  0, 10, 10, 10, 10,  0,  0,
    0,  0, 10, 15, 15, 10,  0,  0,
    0,  0, 10, 15, 15, 10,  0,  0,
    0,  0, 10, 10, 10, 10,  0,  0,
    0,  0,  0,  0,  0,  0,  0,  0,
    0,  0,  0,  0,  0,  0,  0,  0
},
```

Figure 7.
Queen's Basic Value

Queen piece doesn't have a significantly different value while placed on any square. However, just like rook, queen doesn't have any square which would make them in a disadvantage, assuming there's no piece stand on their move range. But a queen have some square where they would get 15 point. This is considered as great value, assuming a queen is at the same time is mostly a main target as capturing a queen will at the same time chipping opponent's power in great value.

```
// King scores White
{    0,  0,  0,  0,  0,  0,  0,  0,
     0,  0,  0,  0,  0,  0,  0,  0,
     0,  0,  0,  0,  0,  0,  0,  0,
     0,  0,  0,  0,  0,  0,  0,  0,
    12,  8,  4,  0,  0,  4,  8, 12,
    16, 12,  8,  4,  4,  8, 12, 16,
    24, 20, 16, 12, 12, 16, 20, 24,
    24, 24, 24, 16, 16,  6, 32, 32},
// King end-game scores White
{   -30, -5,  0,  0,  0,  0,  -5, -30,
     -5,  0,  0,  0,  0,  0,   0,  -5,
      0,  0,  0,  0,  0,  0,   0,   0,
      0,  0,  0,  5,  5,  0,   0,   0,
      0,  0,  0,  5,  5,  0,   0,   0,
      0,  0,  0,  0,  0,  0,   0,   0,
    -10,  0,  0,  0,  0,  0,   0, -10,
    -40,-10, -5, -5, -5, -5, -10, -40}
```

Figure 8.
King's Basic Value

King, logically, will have different purpose between the beginning of the game and toward the end of the game. While the game is near it's end, less pieces will be available, thus the king's struggle to survive is more depended on the position of itself rather than depending on piece's defense.

Despite all that, King still possess some threat. This is mainly because near the game's end the side at disadvantage will most likely reserve to a stalemate option. Which is easier to be done when the king have more squares to move.

Applying Alpha –Beta Algorithm

This is the main part of the engine. Search. cpp file will calculate and weighing whether a value is worthy to be evaluated using Alpha-Beta algorithm. In this matter, we will make the engine operates differently to avoid overlapping.

```
Board =(*GameBoard);
MakeMove(&vMoves[i]);
score=-Negamax(&Board,iDepth+1,!iColor);
if(score>bestscore) bestscore=score;
```

Figure 9
basic scoring

The algorithm that just shown, show us Minimax process of sorting inside the engine. Basically, Alpha-Beta pruning will cut unnecessary node which considered unworthy to be calculated further.

```
If(iDepth>=iMaxDepth)
return EvaluateBoard(GameBoard,iColor);
else sort(vMoves.begin(),vMoves.end());
for(i=0;i<(int)vMoves.size();i++)
{
    Board AuxBoard=(*GameBoard);
    AuxBoard.MakeMove(&vMoves[i]);
    score=-AlphaBeta(&AuxBoard,iDepth+1,-
iBeta,-iAlpha,!iColor);
    if(score>iBeta) return score;
if(score>bestscore)
    {       bestscore=score;
            if(score>iAlpha) iAlpha=score;
}
return bestscore;
```

Figure 10.
Cut off's Logic

Algorithm above briefly explain to us on how armonia chess engine will do it's pruning on unnecessary nodes.

Testing

The testing is done using Arena version 3.5, a GUI chess interface which commonly used in online match. The testing hardware use a dual-core Intel processor and 2GB RAM.

1st match: Harmonia vs Houini 1.5

Harmonia chess engine will be up against Houdini 1.5 on the first test. In the first match, Harmonia will play as white, and Houdini 1.5 will play as black.

Table 1. Opening of 1st Match

No	White : Harmonia	Black: Houdini
1	d4	Nf6
2	Nc3	d5
3	Bf4	e6
4	e3	Bb4
5	Bd3	c5
6	dxc5	

Figure 11.
Position on 1st match after 60. ... Rh7

After 60th move, Harmonia commit a 3 move repetition, hence the game was concluded with a stalemate.

2nd Match: Harmonia vs Hermann

In the 2nd test, the out of 4 plays, none could be finished. In every play, either one of the engine would eventually crash.

Table 2. Opening of 2nd Match

White : Harmonia (175 KB)	Black : Hermann (497 KB)
1.Nc3	Nf6
2.d4	d5
3.e4	

A bad move from Harmonia. Harmonia apparently wasting this pawn for no reason. Hermann evaluate this mistake and make no hesitation as an answer.

Table 3.
Mid-game of 2nd Match

White : Harmonia (175 KB)	Black : Hermann (497 KB)
	Nxe4
Qh5+	g6.

This is the second mistake from Harmonia, a check move without proper preparation.

Figure 12.
Position on 2nd match after 14.Bb4 Be7

After some notable moves, Harmonia crashed itself in midway.

Table 4. End-game of 2nd Match

White : Harmonia (175 KB)	Black: Hermann (497 KB)
Bb5	Bxb4
White crash	

3rd Match: Harmonia vs AnMon

In the last match, Harmonia will be faced with AnMon engine. AnMon is slightly different with the previous engine Harmonia had faced. AnMon using Nalimov database as a playbook and Quiescence algorithm as it's base.

Table 5. Opening of 3rd Match

	White : Harmonia (175 KB)	Black : AnMon (215 KB + Nalimov database 1,2 MB)
1	e4	c6
2	d4	d5

Engine with playbook will most likely playing some particular pattern in the early game, based on the database it has.

Figure 17.
Position on 3rd match after 12. Qb3 Na5

After 12th turn, harmonia and AnMon made a lot of exchange of it's pieces. The condition afterward is as follow.

Table 6. Mid-game of 3rd Match

White : Harmonia (175 KB)	Black : AnMon (215 KB+Nalimov database 1,2 MB)
Rf8	Ke7
Rg8	

Harmonia just wasting a turn by moving it's rook to g8, which supposed to be h8, so that would stop Black king's pursue over the rook.

Figure 19.
Position on 3rd match after 29. Rg8

By this point, Harmonia commit a 3 move repetition which resulted on a stalemate.

CONCLUSION

Two out of 3 matches, Harmonia was able to give 2 stalemate results. On 1st and 3rd match, Harmonia was able to avoid losing the match despite some wrong move it has made.

Table 7. Resource Used in Each of Chess Engine

Engine	Avg % CPU	Memory	Size
Harmonia	51	110.824 KB	175 KB
Hermann	52	138.888 KB	497 KB
Houdini	55	140 055 KB	1,5 MB
Anmon	54	134.444 KB	1,4 MB

From tha given data, it's clear that Harmonia using the least resource compared to another engine given above.

Harmonia's opponent in a match was deliberately chosen with the bigger size, which infer that the engine would have more option in it's calculation. The fact that Harmonia was able to give 2 stalemate out of 3 matches. This conclude that Alpha-Beta pruning algorithm is a solution to cover an engine's limitation with effective calculation.

BIBLIOGRAPHY

Abdelbar, A.M., 2005, Alpha-Beta Pruning and Alth¨ofer's Pathology-Free Negamax Algorithm, *ICCA*.

Dechter, R., 2010, *Rapier: A Chess Engine*.

Elo, A. E., Sloan, S., 2008, *The Rating of Chess Players, Past and Present Paperback*.

Eppstein, D., 1999, *Finding the k shortest paths*, Dept. Information & Computer Science, UC Irvine,

Festa, J., Davino, S., 2013, "IAgo Vs Othello": An artificial intelligence agent playing Reversi. *ceur-ws.org/Vol-1107*, diakses 10 November 2014.

John, F.N. , 1950 *Equilibrium Points in n-Person Games*. Proc. Natl. Acad. Sci. USA .

Lin, Y. C. 2014, The Hierarchical Minimax Theorems, *http://journal.taiwanmathsoc.org.tw*, diakses 10 November 2014.

Raghavan, T.E.S., 1994, *Handbook of Game Theory*, Elsevier Science B.V., , 738.

Reinefield, A., 1983, An Improvement to The Scout Search Tree Algorithm, *ICCA Journal*.